RSAC 2025 Conference

Many Voices.
One Community.

SESSION ID: APP-R02

AI-Powered Fuzzing:

Breaking the Barrier of Auto Vulnerability Discovery

Oliver Chang

Senior Staff Software Engineer
Google

Disclaimer

Presentations are intended for educational purposes only and do not replace independent professional judgment. Statements of fact and opinions expressed are those of the presenters individually and, unless expressly stated to the contrary, are not the opinion or position of RSA Conference LLC or any other co-sponsors. RSA Conference LLC does not endorse or approve, and assumes no responsibility for, the content, accuracy or completeness of the information presented.

Attendees should note that sessions may be audio- or video-recorded and may be published in various media, including print, audio and video formats without further notice. The presentation template and any media capture are subject to copyright protection.

© 2025 RSA Conference LLC or its affiliates. The RSAC and RSAC CONFERENCE logos and other trademarks are proprietary. All rights reserved.

Outline

- Background and Why
- Results
- OSS-Fuzz-Gen V1
- OSS-Fuzz-Gen V2
- Future work

Vulnerability Detection via Fuzzing

Repeatedly executing code with many randomized inputs to find bugs.

Consider fuzzing a function that takes a directory name and `ls` its content*:

```
def list_dir(dir_name):
  """Shows |dir_name| content."""
  os.system('ls ' + dir_name)
```

```
dir_name = 'existing_dir'
```
→
```
file_a file_b subdir_a subdir_b
```

```
dir_name = 'non_existing_dir'
```
→
```
No such file or directory
```

```
dir_name = '; rm -rf /*'
```
→

* Simplified from our past finding: https://security.googleblog.com/2022/09/fuzzing-beyond-memory-corruption.html

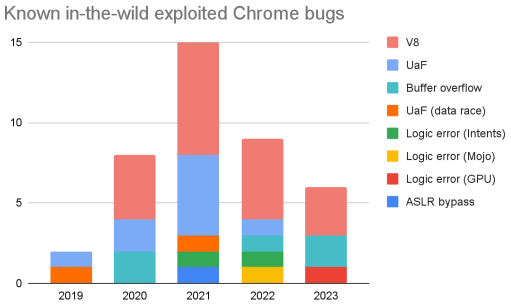
Fuzz target: A wrapper to test a function

```
#include <project_under_test.h>

// Linked with a fuzzing engine, such as AFL++ or libFuzzer which provides
// inputs to this wrapper.
int LLVMFuzzerTestOneInput(const uint8_t* data, size_t size) {
   FuzzedDataProvider provider(data, size);
   auto val1 = provider.ConsumeIntegralInRange<uint16_t>(13000, 16000);
   auto val2 = provider.ConsumeIntegral<uint8_t>();
   auto val3 = provider.ConsumeRemainingBytesAsString();
   DoSomethingInterestingWithMyAPI(val1, val2, val3);
   return 0;
}
```

- Setup the environment,
- Initialize the parameters based on input provided by fuzzing engine.
- Invoke the function-under-test with the randomized parameters.

Memory Corruption: Impact & Exploitation

- C/C++: Memory unsafe
- One of the most severe and widely-exploited vulnerabilities
  - More than half of 0-days tracked by Project Zero



ClusterFuzz

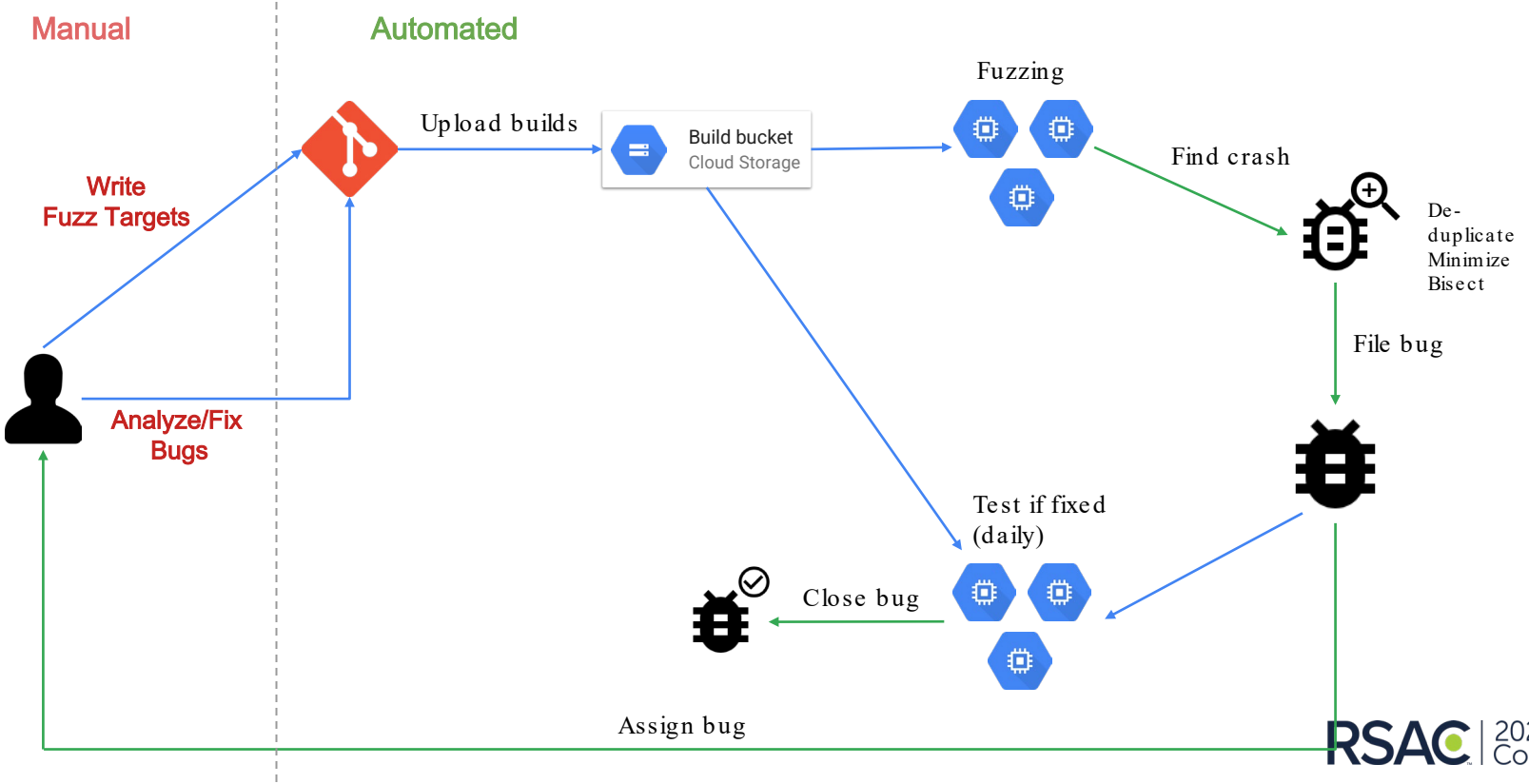
ClusterFuzz:

- Backend for all Google fuzzing infrastructure
- ~150k bugs filed
- Mostly finds memory corruption in C/C++

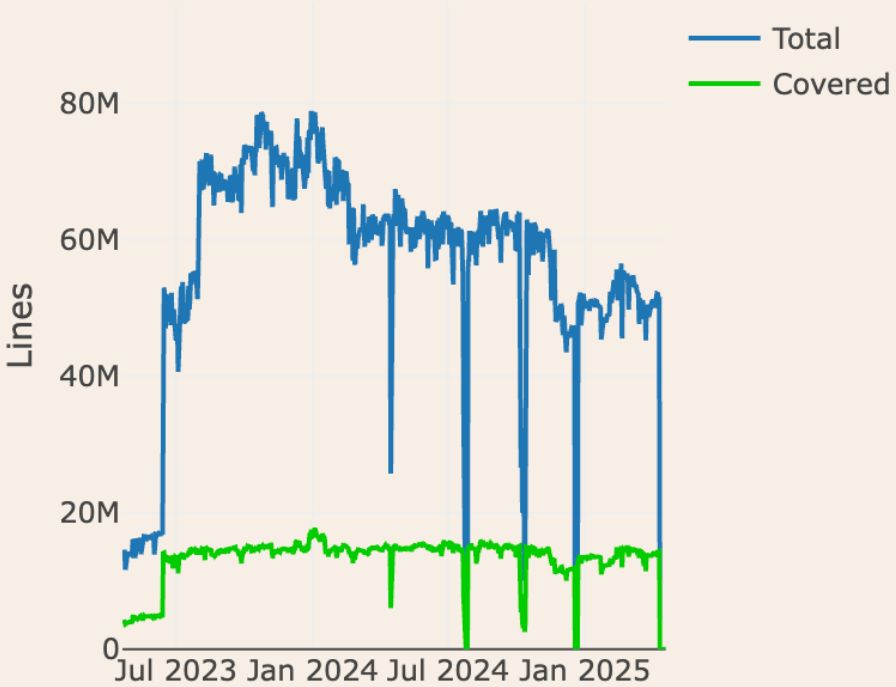
OSS-Fuzz:

- ClusterFuzz for Open Source
- Fuzzing 1200+ critical open source projects (e.g., curl, OpenSSL)

Automate Vulnerability Detection



Why: Fully automate vulnerability discovery (and patching)



Lines of code

~30% avg coverage

Insufficient Fuzz targets

- OSS-Fuzz lacking code coverage for **~70%** of all projects integrated

- Research[1] proves **more/better fuzz targets** is the solution for most of this.
  - Fuzz wider rather than deeper
  - Get an LLM to do this!

[1] https://dl.acm.org/doi/abs/10.1145/3605157.3605177

Requirement	Human-written	OSS-Fuzz-Gen
Fuzzing knowledge	Some required	LLM pretrained
Project domain knowledge	Required	Programmatically supplied in prompts
Time	Hours-Days	Seconds
Cost	$1000+*	<$10

*Based on our OSS-Fuzz rewards program

Results

Results: New Bug Discovery

libplist

> Thanks for reporting. I was about to say, interesting that clusterfuzz didn't catch that one yet, but plist_from_memory isn't part of the harness.

Results: New Bug Discovery

libplist

> Thanks for reporting. I was about to say, interesting that clusterfuzz didn't catch that one yet, but plist_from_memory isn't part of the harness.

cJSON

Note this isn't caught by the existing fuzz target because it enforces the input having a null terminator, which hides this OOB access:

cJSON/fuzzing/cjson_read_fuzzer.c
Line 23 in cb8693b

```
23      if(data[size-1] != '\0') return 0;
```

Results: New Bug Discovery

OpenSSL

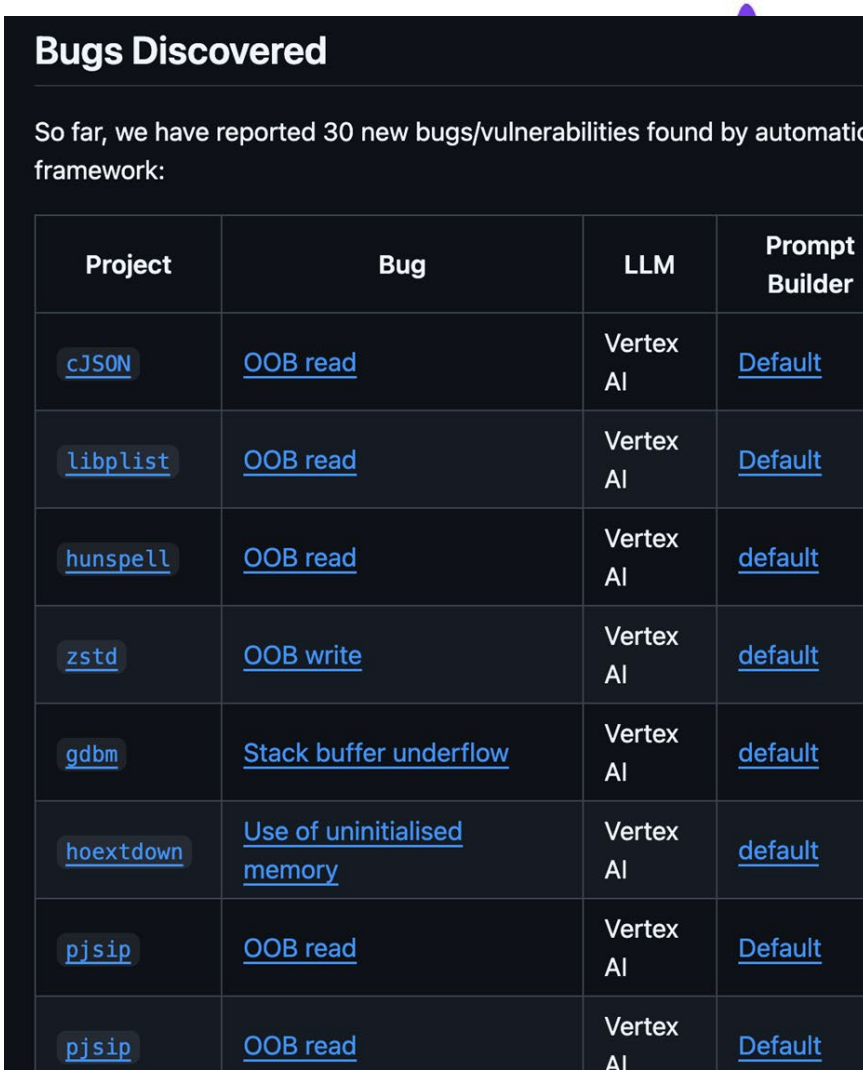
```
OpenSSL Security Advisory [16th October 2024]
=============================================

Low-level invalid GF(2^m) parameters lead to OOB memory access (CVE-2024-9143)
===============================================================================

Severity: Low

Issue summary: Use of the low-level GF(2^m) elliptic curve APIs with untrusted
explicit values for the field polynomial can lead to out-of-bounds memory reads
or writes.

Impact summary: Out of bound memory writes can lead to an application crash or
even a possibility of a remote code execution, however, in all the protocols
involving Elliptic Curve Cryptography that we're aware of, either only "named
curves" are supported, or, if explicit curve parameters are supported, they
specify an X9.62 encoding of binary (GF(2^m)) curves that can't represent
```

Results: New Bug Discovery

30* new vulnerabilities discovered

Including CVE-2024-9143 in OpenSSL

* = limiting factor being human triage (more on that later)



Bugs Discovered

So far, we have reported 30 new bugs/vulnerabilities found by automatic framework:

Project	Bug	LLM	Prompt Builder
cJSON	OOB read	Vertex AI	Default
libplist	OOB read	Vertex AI	Default
hunspell	OOB read	Vertex AI	default
zstd	OOB write	Vertex AI	default
gdbm	Stack buffer underflow	Vertex AI	default
hoextdown	Use of uninitialised memory	Vertex AI	default
pjsip	OOB read	Vertex AI	Default
pjsip	OOB read	Vertex AI	Default

Results: Code Coverage

80% build success rate across 336 C/C++ projects.

2.4% average coverage increase.

Project	Total coverage gain
phmap	98.42%
usbguard	97.62%
onednn	96.67%
avahi	82.06%
pugixml	72.98%
librdkafka	66.88%
casync	66.75%
tomlplusplus	61.06%

Results: New Bug Discovery

These results would never have been obtained from existing human -written fuzz targets.

OSS-Fuzz-Gen V1:

Given a function, generate a fuzz target.

Considerations

- Minimize hallucinations through grounding and feedback
  - We can validate our results automatically to an extend -- this is key
- Replicate typical developer workflow

Prompt: Overview

```
<Goal>

<Guides>

<Context>

<Question>
```

Template for automatic generation

Prompt: Goal

<Goal>

<Guides>

<Context>

<Question>

You are a security testing engineer who wants to write a C++ program to discover memory corruption vulnerabilities in a given function-under-test by executing all lines in it.

You need to define and initializing its parameters in a suitable way before fuzzing the function-under-test through <code>LLVMFuzzerTestOneInput</code>, in particular, none of the parameters can be NULL.

Prompt: Guides

<Goal>	Carefully study the function signature and its parameters, then follow the example problems and solutions to answer the final problem.
<Guides>	Use <code>FuzzedDataProvider</code> to generate these inputs, it is a single-header C++ library that is helpful for splitting a fuzz input into multiple parts of various types.
<Context>	
<Question>	All variables used MUST be declared and initialized. Carefully make sure that the variable and argument types in your code match and compiles successfully. Add type casts to make types match.

Prompt: Context (example solutions)

<Goal>

<Guides>

<Context>

<Question>

Your goal is to write a fuzzing harness for the provided function-under-test signature using <code>LLVMFuzzerTestOneInput</code>. It is important that the provided solution compiles and actually calls the function-under-test specified by the function signature:

<function signature>
BGD_DECLARE(void) gdImageString (gdImagePtr im, gdFontPtr f, int x, int y, unsigned char *s, int color)
</function signature>

<solution>
<An existing human written fuzz target>
</solution>

This is key to making sure we ground the LLM and reduce hallucination

Prompt: Context

<Goal>

<Guides>

<Context>

<Question>

```
< - Function implementation >


XMLNode* XMLElement::ShallowClone( XMLDocument* doc ) const
{
    if ( !doc ) {
        doc = _document;
    }
    XMLText* text = doc->NewText( Value() );
    text->SetCData( this->CData() );
    return text;
}
```

Prompt: Context

<Goal>

<Guides>

<Context>

<Question>

```
< - Project-specific type definition >


typedef struct AvahiDnsPacket {
    size_t size, rindex, max_size, res_size;
    AvahiHashmap *name_table; /* for name compression */
    uint8_t *data;
} AvahiDnsPacket;
```

Prompt: Context

<Goal>

<Guides>

<Context>

<Question>

```
< - Function call sites >


XMLDocument clone;
for(const XMLNode* node=doc.FirstChild(); node; node=node-
>NextSibling()) {
      XMLNode* copy = node->ShallowClone( &clone );
            clone.InsertEndChild( copy );
}
```

All of these are things a human would also need

… but we need to carefully balance including context vs overwhelming LLM context

Prompt: Question

<Goal>

<Guides>

<Context>

<Question>

```
Write a fuzz target for the following function:

void tinyxml2::XMLDocument::Print(tinyxml2::XMLPrinter*) const

<The function's contextual information, e.g.:>
< - Function implementation>
< - Project-specific type definition>
< - Function call sites>

Solution:
```

Functions to fuzz automatically determined by analyzing existing code coverage to identify gaps

https://introspector.oss -fuzz.com/target_oracle

Candidate fuzzing harness

- At this point, we have a candidate fuzzing harness from the LLM
- But…we still need to check it
  - Does it compile?
  - Does it instantly crash?
  - How much coverage does it gain?
- Feedback results to LLM and iterate
  - Very similar to a human developer's process

Feedback: fixing compile errors

```
Below is the error to fix:
The code has the following build issues:
<error>
/src/aom/examples/av1_dec_fuzzer.cc:25:20: warning: variable length arrays in C++
are a Clang extension [-Wvla-cxx-extension]
   25 |   tran_low_t input[rows * cols];
      |                    ^~~~~~~~~~
/src/aom/examples/av1_dec_fuzzer.cc:25:20: note: initializer of 'rows' is not a
constant expression
/src/aom/examples/av1_dec_fuzzer.cc:19:13: note: declared here
   19 |   const int rows = 1 << tx_size_high;
      |             ^

</error>
```

Feedback: fixing compile errors

```
Explanation of changes:

1. **Included necessary headers:** `txfm_common.h` for `TX_CLASS` and `enums.h` for `TX_SIZE`, `DCT_DCT`, and `TxSetType`.  This resolves the unknown type name errors.
2. **Replaced VLAs with `std::vector`:**  Variable Length Arrays (VLAs) are not standard C++ and can cause issues.  Using `std::vector` provides dynamic sizing while remaining standard compliant.  The `.data()` method is used to get a raw pointer to the underlying data for compatibility with the function being tested.
3. **Set `is_hbd`:** The original code was missing the `is_hbd` member initialization in the `TxfmParam` struct.  This has been added and initialized to 0.  Since `bd` is 8, `is_hbd` should be 0 (not high bit depth).
4. **Initialized `eob`:** While not strictly necessary for compilation, initializing `eob` to a sensible default value (like 0) is good practice and can prevent undefined behavior.
5. **Removed `is_inter` and `tx_class`:**  These fields are not part of the `TxfmParam` struct definition provided, so attempting to access them results in a compilation error.  Since the provided definition doesn't include these fields, they have been removed.  If a different version of the `TxfmParam` struct *does* include these fields, the appropriate header should be included.
```

More feedback: fixing runtime errors

```
extern "C" int LLVMFuzzerTestOneInput(const uint8_t
*data, size_t size) {
  libusb_device_handle *handle = nullptr;
  libusb_bos_descriptor *bos_descriptor = nullptr;
  FuzzedDataProvider stream(data, size);
  int ret = libusb_get_bos_descriptor(handle,
      &bos_descriptor);
  if (ret == LIBUSB_SUCCESS) {
    libusb_free_bos_descriptor(bos_descriptor);
  }
  return 0;
}
```

Incorrect in several ways:

1. Executes function with constant parameters.
2. The parameters are also invalid.

* External collaboration with NTU researchers based on their recent studies

Fixing runtime errors

```
extern "C" int LLVMFuzzerTestOneInput(const uint8_t
*data, size_t size) {
  libusb_device_handle *handle = nullptr;
  libusb_bos_descriptor *bos_descriptor = nullptr;
  FuzzedDataProvider stream(data, size);
  int ret = libusb_get_bos_descriptor(handle,
      &bos_descriptor);
  if (ret == LIBUSB_SUCCESS) {
    libusb_free_bos_descriptor(bos_descriptor);
  }
  return 0;
}
```

Immediate crash

```
Stack 0:
#0 0x572144 in libusb_control_transfer /src/libusb/libusb/sync.c:112:27
#1 0x56f7b4 in libusb_get_descriptor /src/libusb/libusb/./libusb.h:2064:9
#2 0x56f7b4 in libusb_get_bos_descriptor /src/libusb/libusb/descriptor.c:857:6
#3 0x56c65c in LLVMFuzzerTestOneInput /src/libusb_fuzzer.cc:12:13
#4 0x43dfc3 in fuzzer::Fuzzer::ExecuteCallback(unsigned char const*, unsigned
long) /src/llvm-project/compiler-rt/lib/fuzzer/FuzzerLoop.cpp:611:15
#5 0x43f374 in
fuzzer::Fuzzer::ReadAndExecuteSeedCorpora(std::__Fuzzer::vector<fuzzer::SizedF
ile, std::__Fuzzer::allocator<fuzzer::SizedFile> >&) /src/llvm-
project/compiler-rt/lib/fuzzer/FuzzerLoop.cpp:804:3
#6 0x43f849 in fuzzer::Fuzzer::Loop(std::__Fuzzer::vector<fuzzer::SizedFile,
std::__Fuzzer::allocator<fuzzer::SizedFile> >&) /src/llvm-project/compiler-
rt/lib/fuzzer/FuzzerLoop.cpp:857:3
#7 0x42eeaf in fuzzer::FuzzerDriver(int*, char***, int (*)(unsigned char
const*, unsigned long)) /src/llvm-project/compiler-
rt/lib/fuzzer/FuzzerDriver.cpp:912:6
#8 0x458502 in main /src/llvm-project/compiler-
rt/lib/fuzzer/FuzzerMain.cpp:20:10
#9 0x7ff2cfb51082 in __libc_start_main (/lib/x86_64-linux-
gnu/libc.so.6+0x24082) (BuildId: eebe5d5f4b608b8a53ec446b63981bba373ca0ca)
#10 0x41f8ed in _start (out/libfuzzer-address-x86_64/libusb_fuzzer+0x41f8ed)
```

Fixing runtime errors

```
# Prompt

Given the following C++ fuzz target
and its fuzing runtime error message.
Edit the source code to fix the
error.

<Fuzz target source code to fix>

<Error summary>

<Crash call-stack + Fuzzing runtime
log>
```

Fixing runtime errors

Prompt

Given the following C++ fuzz target and its fuzing runtime error message. Edit the source code to fix the error.

<Fuzz target source code to fix>

<Error summary>

<Crash call-stack + Fuzzing runtime log>

```
extern "C" int LLVMFuzzerTestOneInput(const uint8_t *data, size_t
size) {
  libusb_context *ctx = nullptr;
  libusb_device_handle *handle = nullptr;
  libusb_bos_descriptor *bos_descriptor = nullptr;
  FuzzedDataProvider stream(data, size);
  int ret = libusb_init(&ctx);
  if (ret != LIBUSB_SUCCESS) {
    return 0;
  }
  handle = libusb_open_device_with_vid_pid(ctx,
    stream.ConsumeIntegral<uint16_t>(),
    stream.ConsumeIntegral<uint16_t>());
  if (!handle) {
    libusb_exit(ctx);
    return 0;
  }
  ret = libusb_get_bos_descriptor(handle, &bos_descriptor);
  if (ret == LIBUSB_SUCCESS) {
    libusb_free_bos_descriptor(bos_descriptor);
  }
  libusb_close(handle);
  libusb_exit(ctx);
  return 0;
}
```

Collecting results at scale

- Code coverage
- Crashes

	Benchmark	Status	Build Rate	Crash Rate	Bugs	Program Counter Coverage	Line Coverage Diff
0	speex `float vbr_analysis(VBRState *, spx_word16_t *, int, int, float)`	Done	20.00	0.00	0	65.89	88.28
1	tarantool `void port_sql_dump_lua(struct port *, struct lua_State *, DW_TAG_enumeration_typeport_dump_lua_mode)`	Done	20.00	0.00	0	75.00	40.74
2	re2 `bool absl::SimpleAtod(string_view, Nonnull<double *>)`	Done	80.00	0.00	0	21.96	26.87
3	tinyobjloader `struct shape_t & tinyobj::shape_t::operator=(struct shape_t *, struct DW_TAG_rvalue_reference_type shape_t)`	Done	50.00	0.00	0	19.94	23.53

e

Learning from Experts

```
extern "C" int LLVMFuzzerTestOneInput(const uint8_t* data, size_t size) {
  std::string data_string(reinterpret_cast<const char*>(data), size);
  tinyxml2::XMLDocument doc;
  doc.Parse(data_string.c_str());
  ----------------------------------
  return 0;
}
```

Fuzz target #5:
+2.04% coverage

```
tinyxml2::XMLNode* node =
doc.FirstChild();
  if (node != nullptr) {
    node->ShallowEqual(node);
}
```

Fuzz target #4:
+2.07% coverage

```
XMLNode *node = doc.FirstChild();
if (node) {
  XMLElement *element = node->ToElement();
  if (element) {
    element->ShallowClone( &doc );
  }
}
```

Fuzz target #1:
+11.12%coverage

```
tinyxml2::XMLPrinter printer;
doc.Print(&printer);
```

Fuzz target #3:
+3.54% coverage

```
XMLElement *root = doc.FirstChildElement( "root" );
if (root == nullptr) {
  root = doc.NewElement( "root" );
  doc.InsertFirstChild( root );
}
XMLElement *child = doc.NewElement( "child" );
root->InsertEndChild( child );
child->SetAttribute( "name", "foo" );
child->SetAttribute( "value", 1 );
```

Fuzz target #2:
+10.25% coverage

```
doc.SaveFile("out.xml",true);
```

OSS-Fuzz-Gen V2: Agents

Primary difference vs V1: Agents

- Let LLM be more “autonomous”
- Provide tools to obtain more context/grounding
  - Rather than us providing them upfront
- More modular architecture / extensibility

Prompt: Overview

```
<Goal>

<Guides>

<Tools and interaction>

<Question>
```

Prompt: Goal

<Goal>

<Guides>

<Tools and interaction>

<Question>

<system>
As a security testing engineer, you must write an `int **LLVMFuzzerTestOneInput**(const uint8_t *data, size_t size)` fuzz target in C++.

Objective: Your goal is to modify an existing fuzz target **`/src/ada-url/fuzz/url_search_params.cc`** to write a minimum fuzz target of a given function-under-test that can build successfully.
</system>

Prompt: Guides

<Goal>

<Guides>

<Tools and interaction>

<Question>

Step 1. Determine the information you need to write an effective fuzz target.

Step 2. Collect information using the Bash tool.
Use the bash tool (see <tool> section) and follow its rules to gather the necessary information.

Step 3. Analyze the function and its parameters.
Understand the function under test by analyzing its source code and documentation

Step 4. Understand initialization requirements.
Identify what is needed to properly initialize the function:

Prompt: Guides

<Goal>

<Guides>

<Tools and interaction>

<Question>

Step 5. Understand Constraints and edge cases.

Step 6: Plan Fuzz Target Implementation.

Step 7: **Write** the fuzz target code.

Step 8 (Optional): **Modify** the Build Script.

Prompt: Examples

<Goal>

<Guides>

<Tools and interaction>

<Question>

```
<tool>
Bash tool Guide
Use the bash tool to investigate files in the fuzz target's build
environment. This will help you understand the project source code, the
function under test, its dependencies, and any compilation
requirements.

<interaction protocols>
1. Execute Bash Command Message Structure:
* Reason for the Command:
  * Explain the reason for running the command.
  * Wrap this explanation within <reason> and </reason> tags.
* Bash Command:
  * Provide the bash command to execute.
  * Wrap the command with <bash> and </bash> tags.
```

Prompt: Examples

<Goal>

<Guides>

<Tools and interaction>

<Question>

<general rules>
1. File Access and Modification Restrictions:
* Allowed Actions:
* View any files and environment variables in the build environment.

* Prohibited Actions:
* Do not modify, rename, or create new files.
* All modifications will not be preserved when building the fuzz target.
</general rules>

<tool guidelines>
1. Purposeful Commands:
* Each bash command should have a clear purpose related to your investigation toward the final goals.

2. Careful Interpretation:
* Analyze the output of each command thoroughly to inform your next steps.
* Keep notes of important findings that will help in modifying the fuzz target and build script.
…

Prompt: Examples

<Goal>

<Guides>

<Tools and interaction>

<Question>

```
<example usages>

<reason>
To find the definition of `my_function` in the project directory and
understand its implementation details.
</reason>
<bash>
grep -rn 'my_function(' /src/project/
</bash>

<reason>
To identify the header files in the project directory that declare
`my_function`.
</reason>
<bash>
grep -rn 'my_function' /src/project/ --include=*.h
</bash>
```

Prompt: Task

<Goal>

<Guides>

<Tools and interaction>

<Task>

```
<task>
Your goal is to write a fuzzing harness for the provided function-
under-test signature using <code>LLVMFuzzerTestOneInput</code>. It is
important that the provided solution compiles and actually calls the
function-under-test specified by the function signature:

<function signature>
struct url ada::url
ada::parser::parse_url<ada::parse_url<ada::url>(string_view, const
struct url *)
</function signature>

</task>
```

```
+ clang++ -O1 -fno-omit-frame-pointer -gline-tables-only -Wno-error=enum-constexpr-conversion -Wno-error=incompatible-function-pointer-types -Wno-error=deprecated-declarations -Wno-error=vla-cxx-extension -c -o /tmp/verify_name_match_normalizename_fuzzer.cc.o -I/src/boringssl/include -I/src/boringssl/fuzz/ -I/src/boringssl/crypto/ -I/src/boringssl/crypto/fipsmodule/ -I/src/boringssl/crypto/fipsmodule/bcm_interface.h -I/src/boringssl/crypto/fipsmodule/bcm_public.h
```

```
<bash>
cat /src/boringssl/include/openssl/pool.h
</bash>
<return code>
0
</return code>
<stdout>
// Copyright 2016 The BoringSSL Authors
//
// Licensed under the Apache License, Version 2.0 (the "License");
// you may not use this file except in compliance with the License.
// You may obtain a copy of the License at
//
//     https://www.apache.org/licenses/LICENSE-2.0
//
// Unless required by applicable law or agreed to in writing, software
// distributed under the License is distributed on an "AS IS" BASIS,
// WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
// See the License for the specific language governing permissions and
// limitations under the License.

#ifndef OPENSSL_HEADER_POOL_H
#define OPENSSL_HEADER_POOL_H

#include <openssl/base.h>   // IWYU pragma: export
```

```
<conclusion>
The `CRYPTO_BUFFER_new` function is indeed declared in `/src/boringssl/include/openssl/pool.h`. The fuzz target was missing this
</conclusion>
<fuzz target>
// Copyright 2023 The Chromium Authors
//
// Licensed under the Apache License, Version 2.0 (the "License");
// you may not use this file except in compliance with the License.
// You may obtain a copy of the License at
//
//     https://www.apache.org/licenses/LICENSE-2.0
//
// Unless required by applicable law or agreed to in writing, software
// distributed under the License is distributed on an "AS IS" BASIS,
// WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
// See the License for the specific language governing permissions and
// limitations under the License.

#include <fuzzer/FuzzedDataProvider.h>

#include <cstddef>
#include <cstdint>
#include <optional>
#include <string>

#include "/src/boringssl/pki/ocsp.h"
#include "/src/boringssl/pki/parsed_certificate.h"
#include <openssl/pool.h>
```

More tools

- Debuggers (LLDB / GDB)
- Source code indexer
- Code coverage
- And more..

Results of V2 vs V1

- Agentization led to 56.9% -> **80+%** compilation rate

What about accuracy?

- We try to ensure that generated fuzz targets are correct
  - Ensuring it compiles and contributes more code coverage
  - Doesn't instantly crash
- But these are often still wrong
  - Misuses of APIs (e.g. passing a non-null terminated string to API expected null terminated string)
  - Incorrect initialization
  - Bugs in the fuzz target itself

What about accuracy?

- Resulting crashes found need to be **scrutinized by a human** to determine if they're legitimate vulnerabilities
- This is why we've **only** found and reported 30 vulnerabilities
- **Crashes found by generated fuzz harnesses**

… 1000s of untriaged crashes

	Project	Benchmark
1	ada-url	output-ada-url-_zn3ada6parser9parse_urlins_3urleeet_nst3__117basic_string_viewicns4_11ch
2	ada-url	output-ada-url-_zn3ada6parser9parse_urlins_3urleeet_nst3__117basic_string_viewicns4_11ch
3	ada-url	output-ada-url-_zn3ada6parser9parse_urlins_3urleeet_nst3__117basic_string_viewicns4_11ch
4	astc-encoder	output-astc-encoder-_z10encode_ise12quant_methodjpkhphj-01
5	astc-encoder	output-astc-encoder-_z10encode_ise12quant_methodjpkhphj-02
6	astc-encoder	output-astc-encoder-_z10encode_ise12quant_methodjpkhphj-03

Crash triager

Given the following crash report, fuzz driver code and relevant project function code, analyze the cause of the crash.

First, only answer “Crash is caused by bug in fuzz driver” or “Crash is caused by bug in project”. Second, offer a succinct and to-the-point analysis.

Below is crash report:
`<log>...</log>`

Below is driver code:
`<code>..</code>`

Below is relevant project function code:
`<code>..</code>`

Crash is caused by **bug in fuzz driver.**

Analysis:

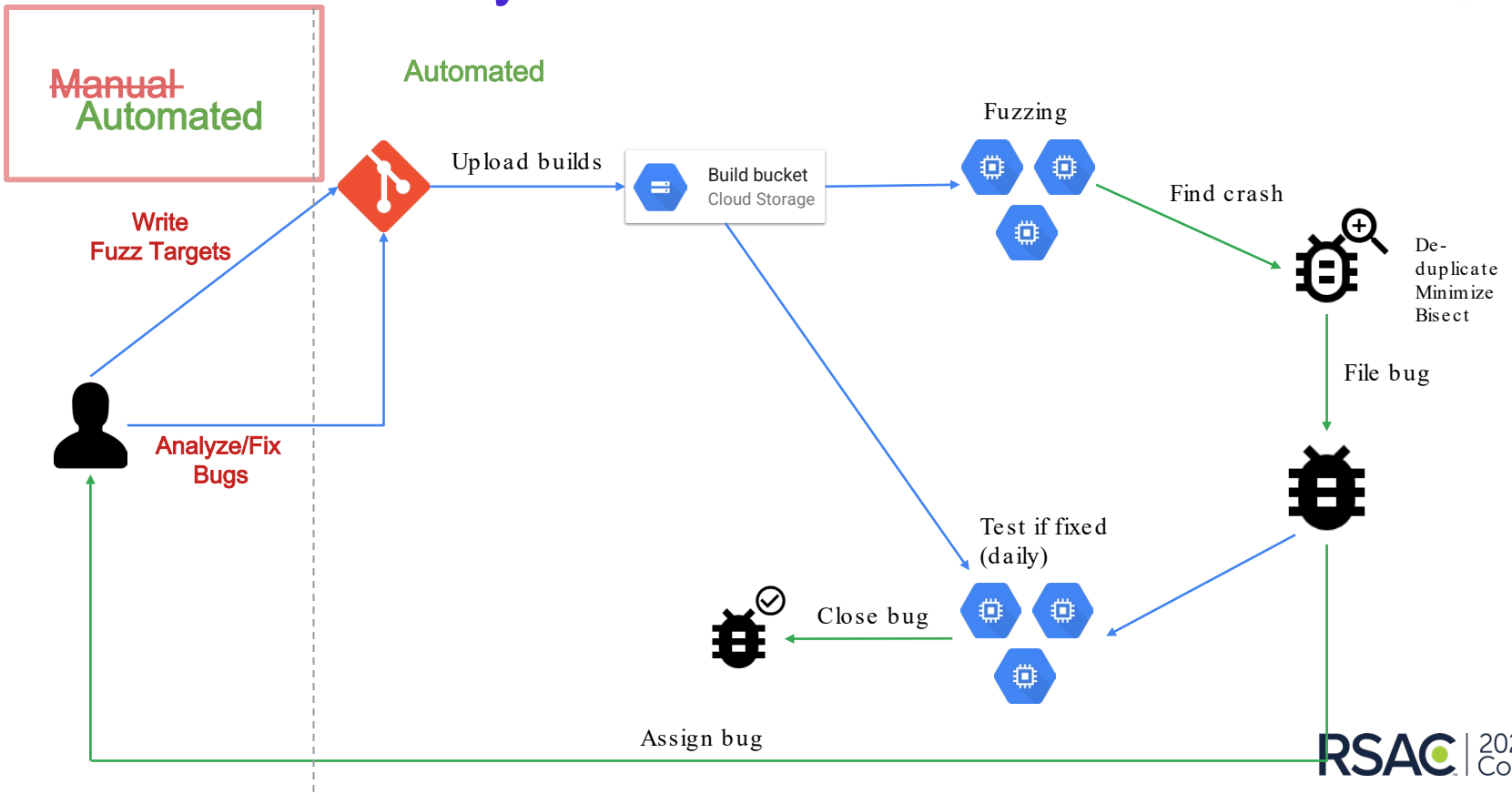
The crash report points to a stack-use-after-scope issue in `LLVMFuzzerTestOneInput`. The problematic line is 44, where `snappy::CompressFromIOVec` is called. The issue stems from passing `iovecs[i].iov_base`, which points to the `c_str()` of `fuzzed_string`, as an argument. However, `fuzzed_string` is a local variable within the loop, and its memory is freed after each iteration, leading to a dangling pointer and the subsequent crash when accessed by `snappy::CompressFromIOVec`.

Future: Agentize, Additional tools: Source code browser + Debugger

Remaining work

- More tools
  - Debuggers (LLDB / GDB)
  - Source code indexer
  - Code coverage
  - And more..
- More agents
  - Crash triager
  - Coverage enhancer
  - Target selection
- Step-by-step reasoning (“thinking”)
- Goal: have this **work reliably well for any arbitrary software project**

Automate Vulnerability Detection



Goal: Fully automate vulnerability discovery (and patching)

Apply: Check out

- Other similar domains?
- All of our work is open source
  - Will soon be integrated into OSS-Fuzz
- https://github.com/google/oss-fuzz-gen
- https://github.com/google/oss-fuzz